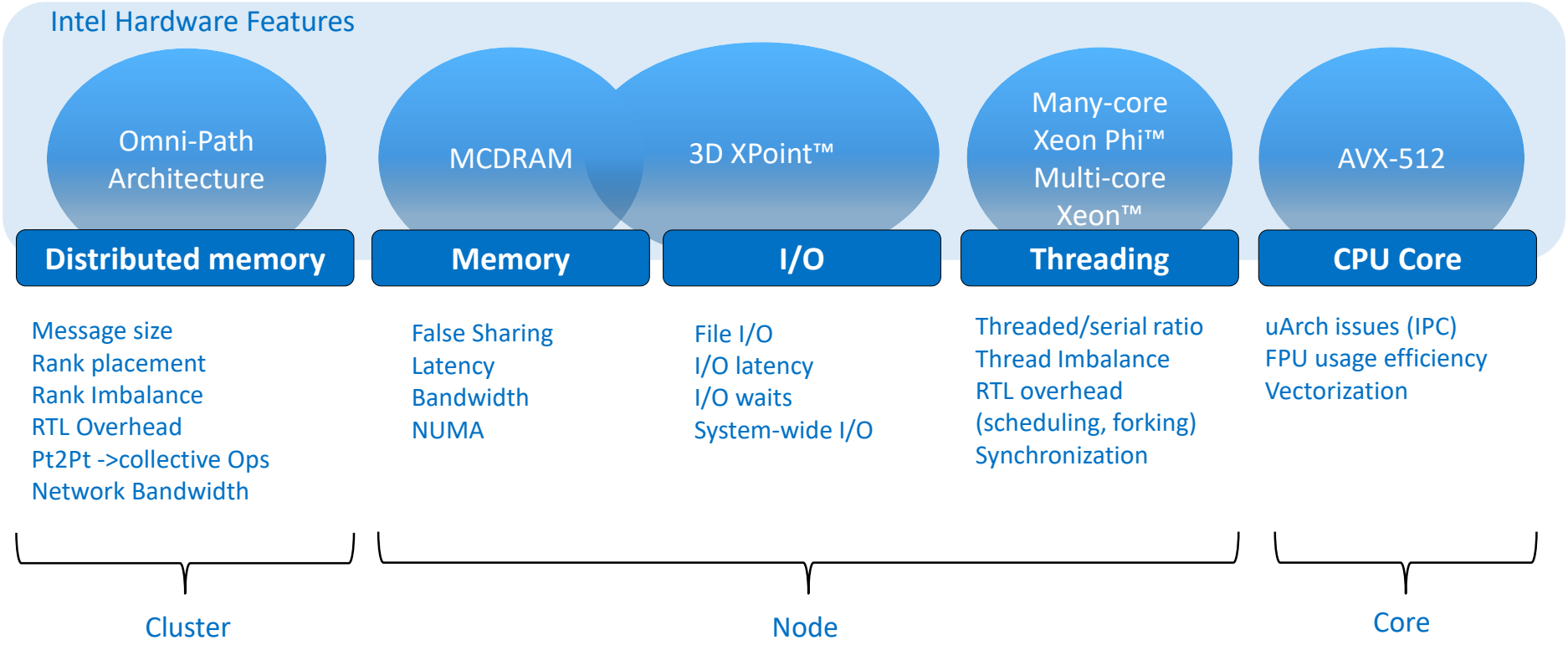
INTEL® VTUNE™ AMPLIFIER'S APPLICATION PERFORMANCE SNAPSHOT:

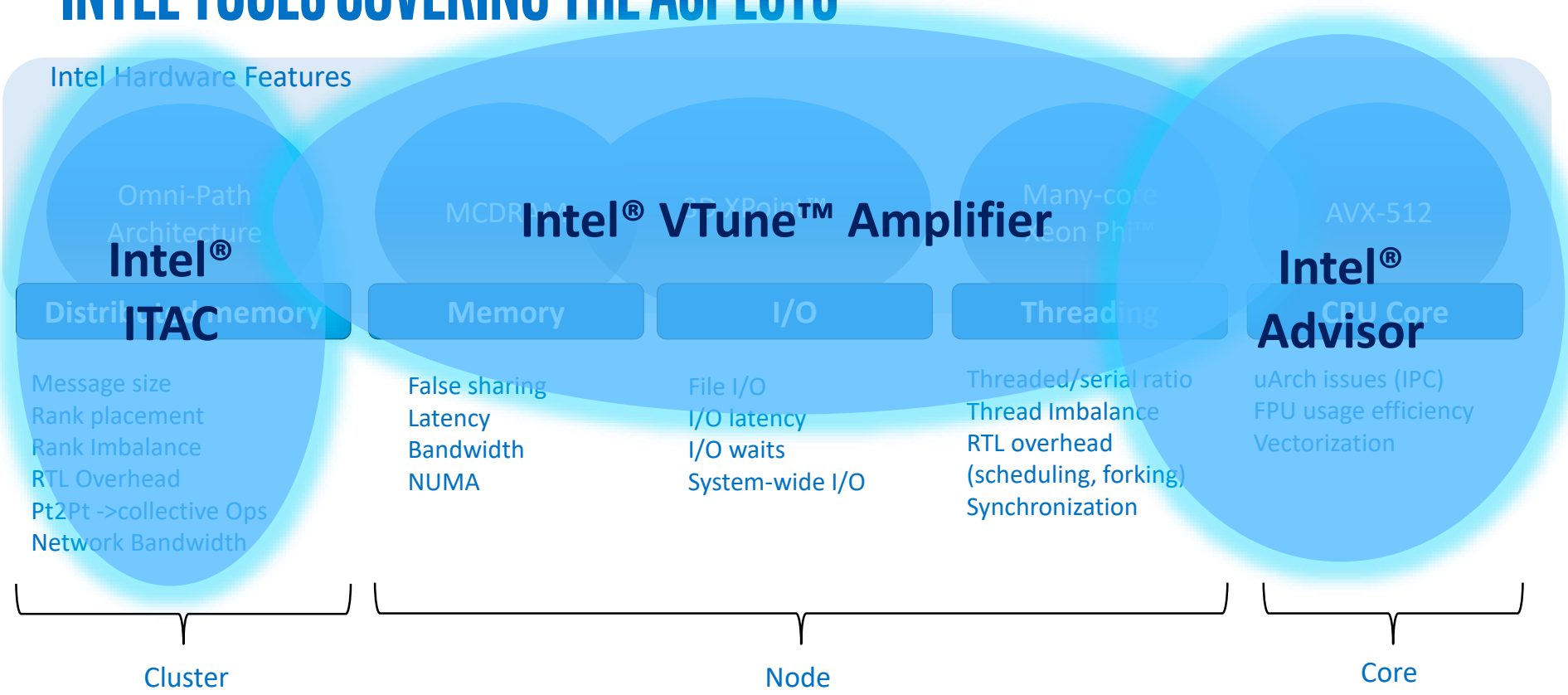
PERFORMANCE OVERVIEW AT SCALE

Dmitry Prohorov,
VTune HPC Lead

ASPECTS OF HPC/THROUGHPUT APPLICATION PERFORMANCE



INTEL TOOLS COVERING THE ASPECTS

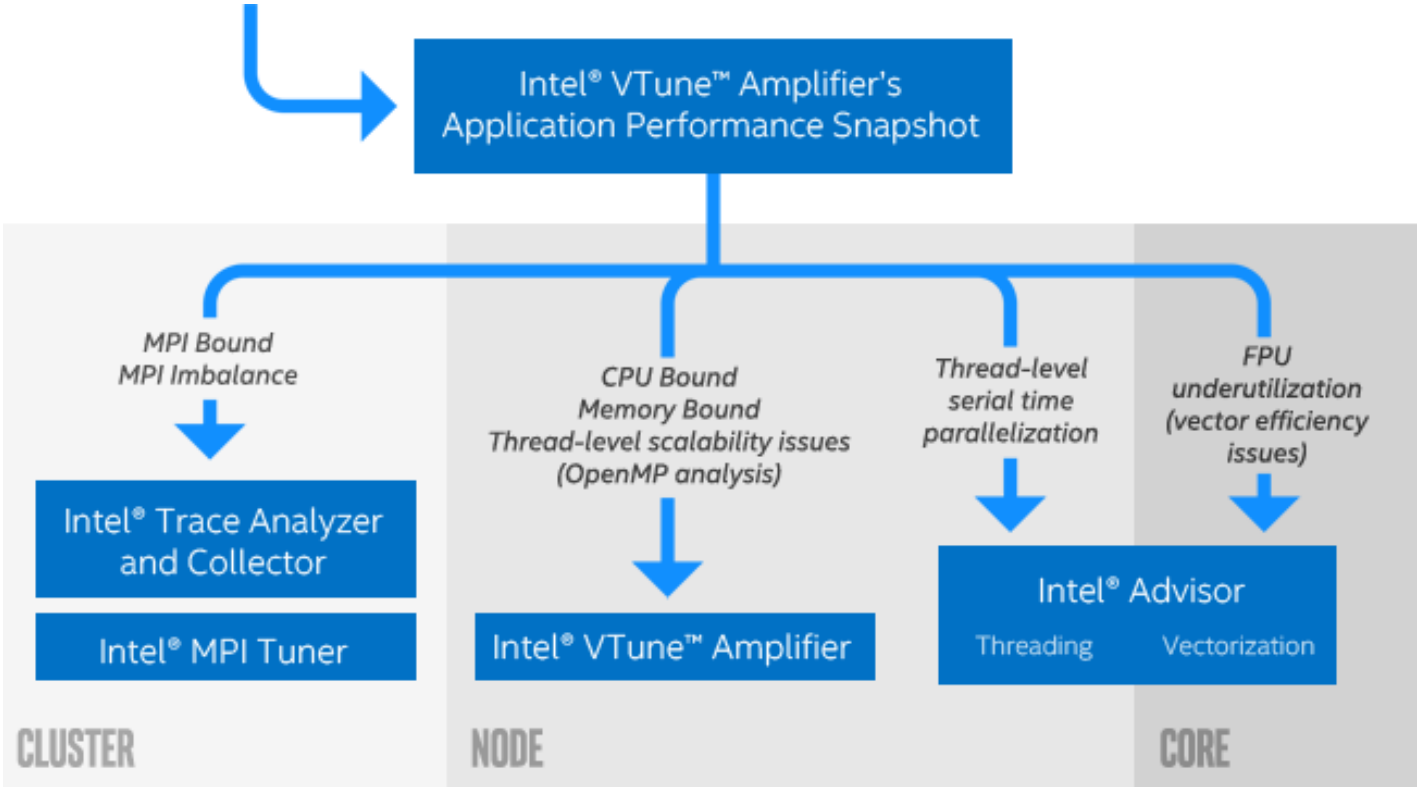


BEFORE DIVING INTO A PARTICULAR TOOL …

- How to assess that I have **potential in performance** tuning?
- **Which tool** should I use first?
- What to use on **large scale** avoiding being overwhelmed with huge trace size, post processing time and collection overhead?
- How to **quickly** evaluate environment settings or incremental code changes?
- Answer:

  Use VTune Amplifier's Application Performance Snapshot 2018

PERFORMANCE OPTIMIZATION WORKFLOW BASED ON APS



APPLICATION PERFORMANCE SNAPSHOT AT A GLANCE (1/2)

- High-level **overview** of application performance
  - Detailed reports on MPI statistics
- Primary optimization areas and **next steps** in analysis with deep tools
- **Easy** to install, run, explore results with CL or HTML reports
  - No driver installation required working through perf
  - If SEP driver is available - will be additional advantage
- Multiple methods to obtain
  - Part of Intel® Parallel Studio XE, VTune Amplifier standalone
  - Separate **free** download (110Mb) from APS web page
    - https://software.intel.com/sites/products/snapshots/application-snapshot/

APPLICATION PERFORMANCE SNAPSHOT AT A GLANCE (2/2)

- **Low** collection overhead – 5-10%
  - HW counters – counting mode only, no overtime
  - MPI and OpenMP tracing - trace aggregation in runtime, no overtime
    - Trace levels to collect more MPI details (potentially for the cost of overhead)
  - Ability to choose either tracing or HW counting in the case of interest in particular metric subset and avoid overhead (--collection-mode option)
- **Scales** to large jobs
  - Tested and worked on 32K ranks
  - Trace size on default statistics level – 4Kb per rank

APS WORKFLOW

Setup Environment

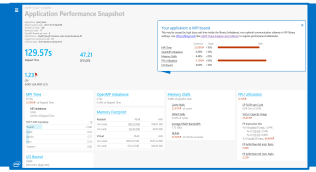
- >source <APS_Install_dir>/apsvars.sh

Run Application

- >aps <application and args>
- MPI: >mpirun <mpi options> aps <application and args>

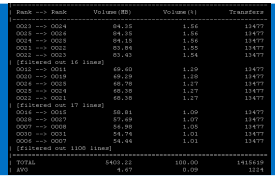
Generate Report on Result Folder

- >aps –report <result folder>

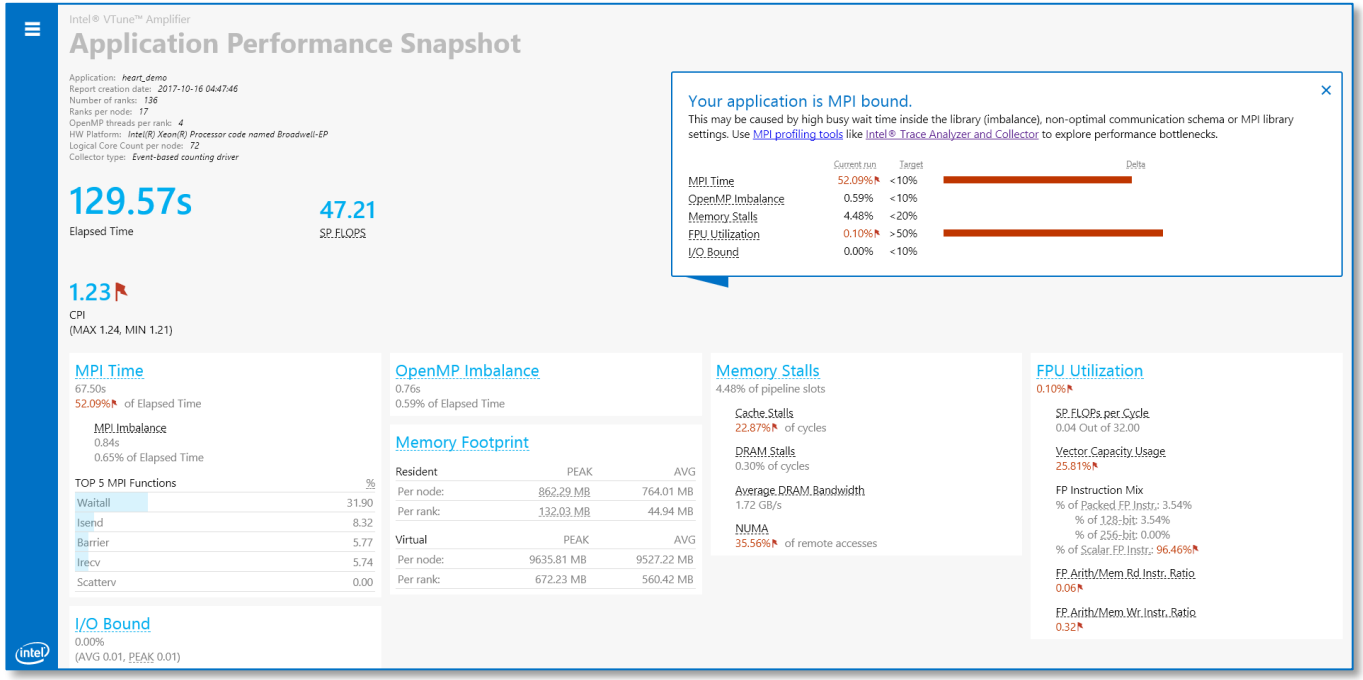


Generate CL reports with detailed MPI statistics on Result Folder

- aps-report –<option> <result folder>



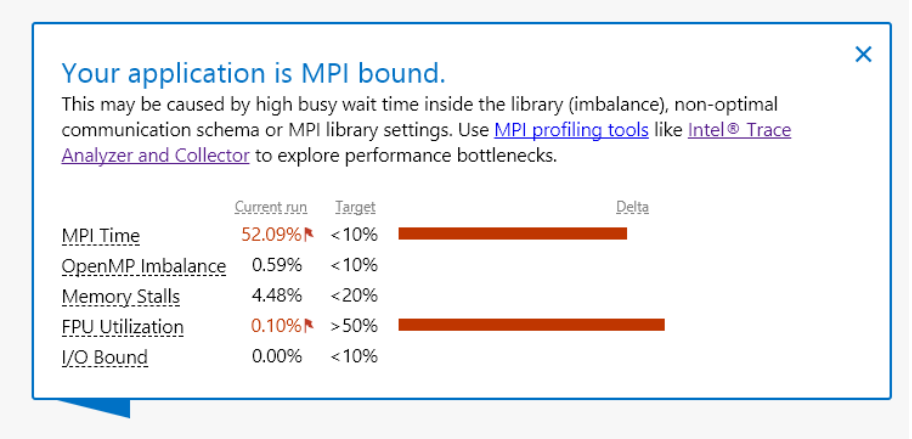
APS HTML REPORT



Report examples (press the links to play): MPI Bound, Memory Bound, OpenMP imbalance

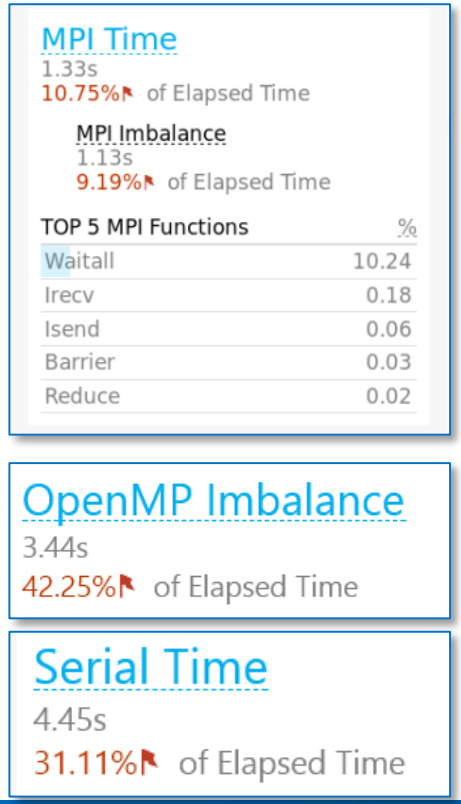
APS HTML REPORT BREAKDOWN - OVERVIEW

- Overview shows all areas and relative impact on code performance
- Provides recommendation for next step in performance analysis
- “X” collapses the summary, removing the flags (objective numbers only)



APS HTML REPORT BREAKDOWN – PARALLEL RUNTIMES

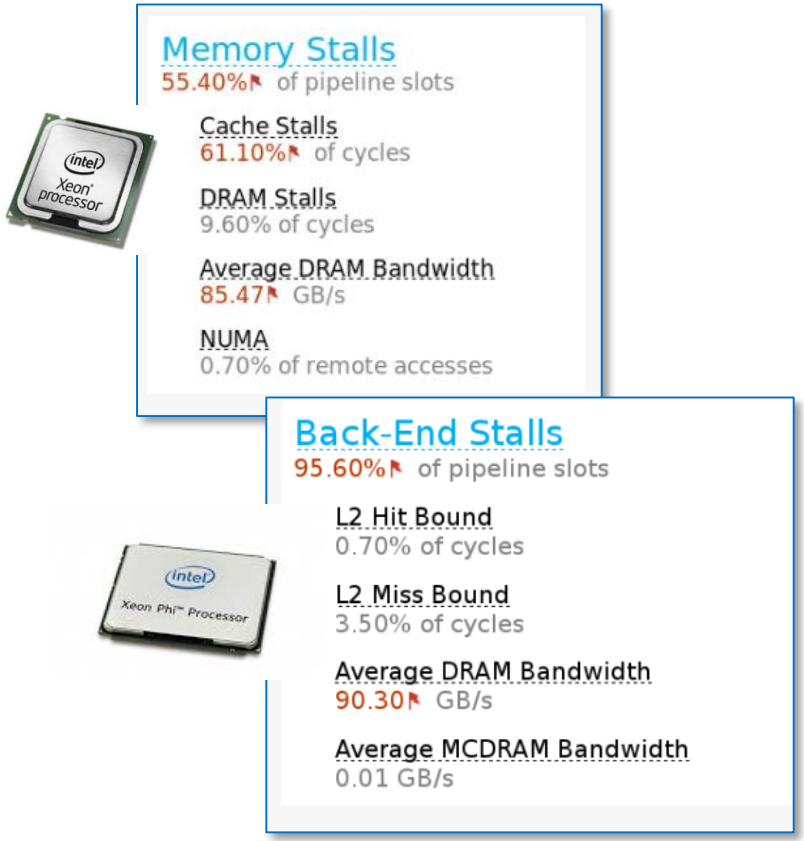
- MPI Time
  - How much time was spent in MPI calls
  - Averaged by ranks with % of Elapsed time
  - Available for MPICH-based MPIs
- MPI Imbalance
  - Unproductive time spent in MPI library waiting for data
  - Available for Intel MPI
- OpenMP Imbalance
  - Time spent at OpenMP Synchronization Barriers normalized by number of threads
  - Available for Intel OpenMP
- Serial time
  - Time spend outside OpenMP regions
  - Available for Intel OpenMP, shared memory applications only



APS HTML REPORT BREAKDOWN – MEMORY ACCESS

- Memory stalls measurement with breakdown by cache and DRAM
- Average DRAM Bandwidth*
- NUMA ratio
- KNL:
  - back-end stalls with L2-demand access efficiency
  - Average DRAM AND MCDRAM Bandwidth*

*Average DRAM and MCDRAM bandwidth collection is available with Intel driver or perf system wide monitoring enabled on a system



APS HTML REPORT BREAKDOWN – VECTORIZATION

- FPU Utilization based on HW-event statistics with
  - Breakdown by vector/scalar instructions
  - Floating point vs memory instruction ratio

- KNL: SIMD Instr. per Cycle
  - Scalar vs. vectorized instructions

FPU Utilization
0.80%

SP FLOPs per Cycle
0.24 Out of 32.00

Vector Capacity Usage
49.90%

FP Instruction Mix
% of Packed FP Instr.: 99.70%
% of 128-bit: 99.70%
% of 256-bit: 0.00%
% of Scalar FP Instr.: 0.30%

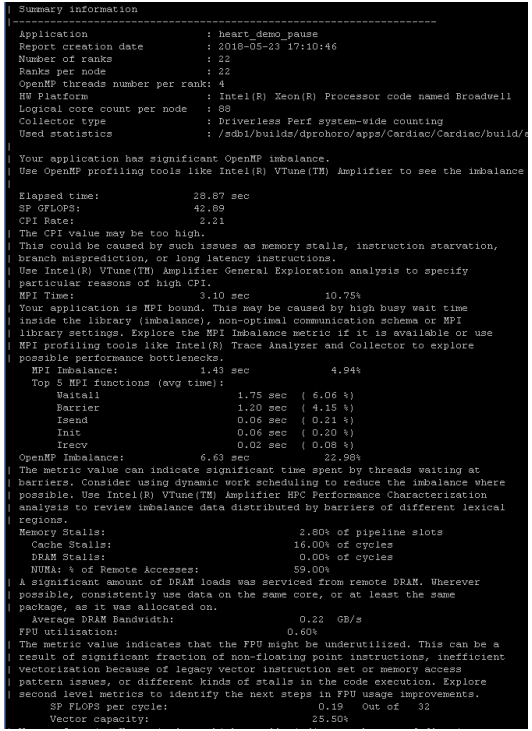
FP Arith/Mem Rd Instr. Ratio
0.41

FP Arith/Mem Wr Instr. Ratio
1.71

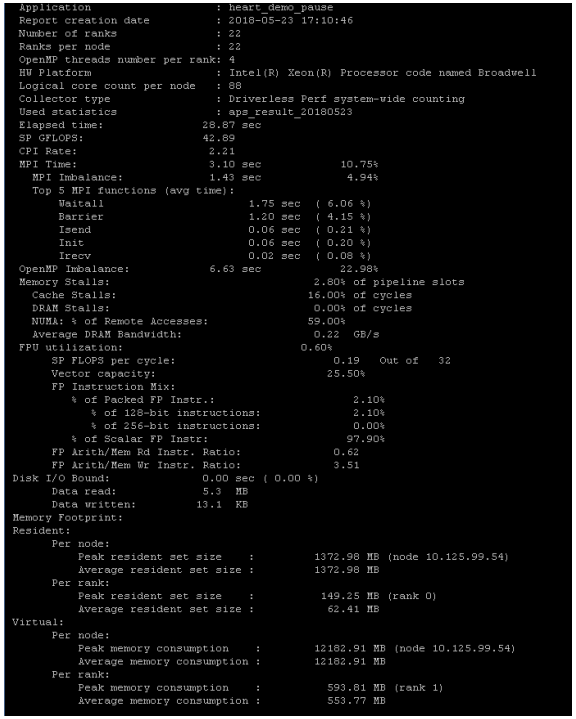
SIMD Instr. per Cycle
0.08

FP Instruction Mix
% of Packed SIMD Instr.: 67.60%
% of Scalar SIMD Instr.: 32.40%

APS COMMAND LINE REPORTS - SUMMARY



```
| Summary information
|--------------------------------------------------------------------
  Application                : heart_demo_pause
  Report creation date       : 2018-05-23 17:10:46
  Number of ranks            : 22
  Ranks per node             : 22
  OpenMP threads number per rank: 4
  HW Platform                : Intel(R) Xeon(R) Processor code named Broadwell
  Logical core count per node   : 88
  Collector type             : Driverless Perf system-wide counting
  Used statistics            : /sdb1/builds/dprohoro/apps/Cardiac/Cardiac/build/s
|
| Your application has significant OpenMP imbalance.
| Use OpenMP profiling tools like Intel(R) VTune(TM) Amplifier to see the imbalance
|
  Elapsed time:              28.87 sec
  SP GFLOPS:                 42.89
  CPI Rate:                   2.21
| The CPI value may be too high.
| This could be caused by such issues as memory stalls, instruction starvation,
| branch misprediction, or long latency instructions.
| Use Intel(R) VTune(TM) Amplifier General Exploration analysis to specify
| particular reasons of high CPI.
  MPI Time:                  3.10 sec            10.75%
| Your application is MPI bound. This may be caused by high busy wait time
| inside the library (imbalance), non-optimal communication schema or MPI
| library settings. Explore the MPI Imbalance metric if it is available or use
| MPI profiling tools like Intel(R) Trace Analyzer and Collector to explore
| possible performance bottlenecks.
    MPI Imbalance:           1.43 sec             4.94%
    Top 5 MPI functions (avg time):
        Waitall                      1.75 sec  ( 6.06 %)
        Barrier                      1.20 sec  ( 4.15 %)
        Isend                        0.06 sec  ( 0.21 %)
        Init                         0.06 sec  ( 0.20 %)
        Irecv                        0.02 sec  ( 0.08 %)
  OpenMP Imbalance:          6.63 sec            22.98%
| The metric value can indicate significant time spent by threads waiting at
| barriers. Consider using dynamic work scheduling to reduce the imbalance where
| possible. Use Intel(R) VTune(TM) Amplifier HPC Performance Characterization
| analysis to review imbalance data distributed by barriers of different lexical
| regions.
  Memory Stalls:                              2.80% of pipeline slots
    Cache Stalls:                            16.00% of cycles
    DRAM Stalls:                              0.00% of cycles
    NUMA: % of Remote Accesses:              59.00%
| A significant amount of DRAM loads was serviced from remote DRAM. Wherever
| possible, consistently use data on the same core, or at least the same
| package, as it was allocated on.
    Average DRAM Bandwidth:                   0.22  GB/s
  FPU utilization:                           0.60%
| The metric value indicates that the FPU might be underutilized. This can be a
| result of significant fraction of non-floating point instructions, inefficient
| vectorization because of legacy vector instruction set or memory access
| pattern issues, or different kinds of stalls in the code execution. Explore
| second level metrics to identify the next steps in FPU usage improvements.
      SP FLOPS per cycle:                     0.19   Out of   32
      Vector capacity:                       25.50%
```



```
  Application                : heart_demo_pause
  Report creation date       : 2018-05-23 17:10:46
  Number of ranks            : 22
  Ranks per node             : 22
  OpenMP threads number per rank: 4
  HW Platform                : Intel(R) Xeon(R) Processor code named Broadwell
  Logical core count per node   : 88
  Collector type             : Driverless Perf system-wide counting
  Used statistics            : aps_result_20180523
  Elapsed time:              28.87 sec
  SP GFLOPS:                 42.89
  CPI Rate:                   2.21
  MPI Time:                  3.10 sec            10.75%
    MPI Imbalance:           1.43 sec             4.94%
    Top 5 MPI functions (avg time):
        Waitall                      1.75 sec  ( 6.06 %)
        Barrier                      1.20 sec  ( 4.15 %)
        Isend                        0.06 sec  ( 0.21 %)
        Init                         0.06 sec  ( 0.20 %)
        Irecv                        0.02 sec  ( 0.08 %)
  OpenMP Imbalance:          6.63 sec            22.98%
  Memory Stalls:                              2.80% of pipeline slots
    Cache Stalls:                            16.00% of cycles
    DRAM Stalls:                              0.00% of cycles
    NUMA: % of Remote Accesses:              59.00%
    Average DRAM Bandwidth:                   0.22  GB/s
  FPU utilization:                           0.60%
      SP FLOPS per cycle:                     0.19   Out of   32
      Vector capacity:                       25.50%
      FP Instruction Mix:
         % of Packed FP Instr.:                2.10%
            % of 128-bit instructions:         2.10%
            % of 256-bit instructions:         0.00%
         % of Scalar FP Instr:                97.90%
      FP Arith/Mem Rd Instr. Ratio:            0.62
      FP Arith/Mem Wr Instr. Ratio:            3.51
Disk I/O Bound:              0.00 sec ( 0.00 %)
      Data read:             5.3  MB
      Data written:         13.1  KB
Memory Footprint:
Resident:
      Per node:
          Peak resident set size    :      1372.98 MB (node 10.125.99.54)
          Average resident set size :      1372.98 MB
      Per rank:
          Peak resident set size    :       149.25 MB (rank 0)
          Average resident set size :        62.41 MB
Virtual:
      Per node:
          Peak memory consumption   :     12182.91 MB (node 10.125.99.54)
          Average memory consumption :     12182.91 MB
      Per rank:
          Peak memory consumption   :       593.81 MB (rank 1)
          Average memory consumption :      553.77 MB
```

Tip:

>aps –report=<my_result_dir> | grep –v “|”

eliminating verbose descriptions

APS COMMAND LINE REPORTS – ADVANCED MPI STATISTICS

aps-report [keys] [options] <result>

[keys] – what to show
--functions
--mpi_time_per_rank
--collop_time_per_rank
--message_sizes
--transfers_per_communication
--transfers_per_rank
--node_to_node
--transfers_per_function
--communicators_list

[options] – how to show
--rank
--comm_id
--details
--communicators
--volume_threshold
--time_threshold
--number_of_lines
--no_filters
--communicators_list
--format

See descriptions with >aps-report command

APS COMMAND LINE REPORTS – ADVANCED MPI STATISTICS (1/3) REPORT EXAMPLES

- MPI Time per rank
  - aps-report –-mpi_time_per_rank <result>

MPI Time per Rank

Rank	LifeTime(sec)	MPI Time(sec)	MPI Time(%)	Imbalance(sec)	Imbalance(%)
0007	72.52	14.31	19.74	4.84	6.67
0004	72.53	11.57	15.96	3.26	4.50
0005	72.52	11.40	15.72	3.20	4.42
0006	72.51	11.11	15.32	3.17	4.37
0000	72.49	11.08	15.29	4.33	5.97
0001	72.52	10.95	15.10	3.01	4.15
0002	72.49	10.79	14.88	2.57	3.55
0003	72.50	10.64	14.68	2.50	3.45
TOTAL	580.07	91.86	15.84	26.88	4.63
AVG	72.51	11.48	15.84	3.36	4.63

APS COMMAND LINE REPORTS – ADVANCED MPI STATISTICS (2/3)

- Message Size Summary by all ranks
  - aps-report –-message_sizes <result>

Message Sizes summary for all ranks

Message size(B)	Volume(MB)	Volume(%)	Transfers	Time(sec)	Time(%)
8	1.49	0.09	195206	27.79	37.93
176	0.41	0.02	2420	27.67	37.78
4	0.00	0.00	1150	15.55	21.22
100264	115.89	6.94	1212	0.27	0.37
98400	113.74	6.81	1212	0.19	0.26
66256	38.29	2.29	606	0.17	0.23
[filtered out 57 lines]					
TOTAL	1670.60	100.00	265160	73.25	100.00

APS COMMAND LINE REPORTS – ADVANCED MPI STATISTICS (3/3)

- Data Transfers for Rank-to-Rank Communication
  - aps-report –-transfers_per_communication <result>

Rank --> Rank	Volume(MB)	Volume(%)	Transfers
0023 --> 0024	84.35	1.56	13477
0025 --> 0026	84.35	1.56	13477
0024 --> 0025	84.15	1.56	13477
0021 --> 0022	83.84	1.55	13477
0022 --> 0023	83.43	1.54	13477
[filtered out 16 lines]			
0012 --> 0011	69.60	1.29	13477
0020 --> 0019	69.29	1.28	13477
0026 --> 0025	68.78	1.27	13477
0025 --> 0024	68.38	1.27	13477
0022 --> 0021	68.38	1.27	13477
[filtered out 17 lines]			
0016 --> 0015	58.81	1.09	13477
0028 --> 0027	57.69	1.07	13477
0007 --> 0008	56.98	1.05	13477
0030 --> 0031	54.74	1.01	13477
0006 --> 0007	54.44	1.01	13477
[filtered out 1108 lines]			
TOTAL	5403.22	100.00	1415619
AVG	4.67	0.09	1224

COLLECTION CONTROL API

Since 2018 Update 2

- To measure a particular application phase or exclude initialization/finalization phases use:

  MPI:

  - Pause: MPI_Pcontrol(0)
  - Resume: MPI_Pcontrol(1)

  MPI or Shared memory applications:

  - Pause: __itt_pause()
  - Resume: __itt_resume()
    - See how to configure the build of your application to use itt API

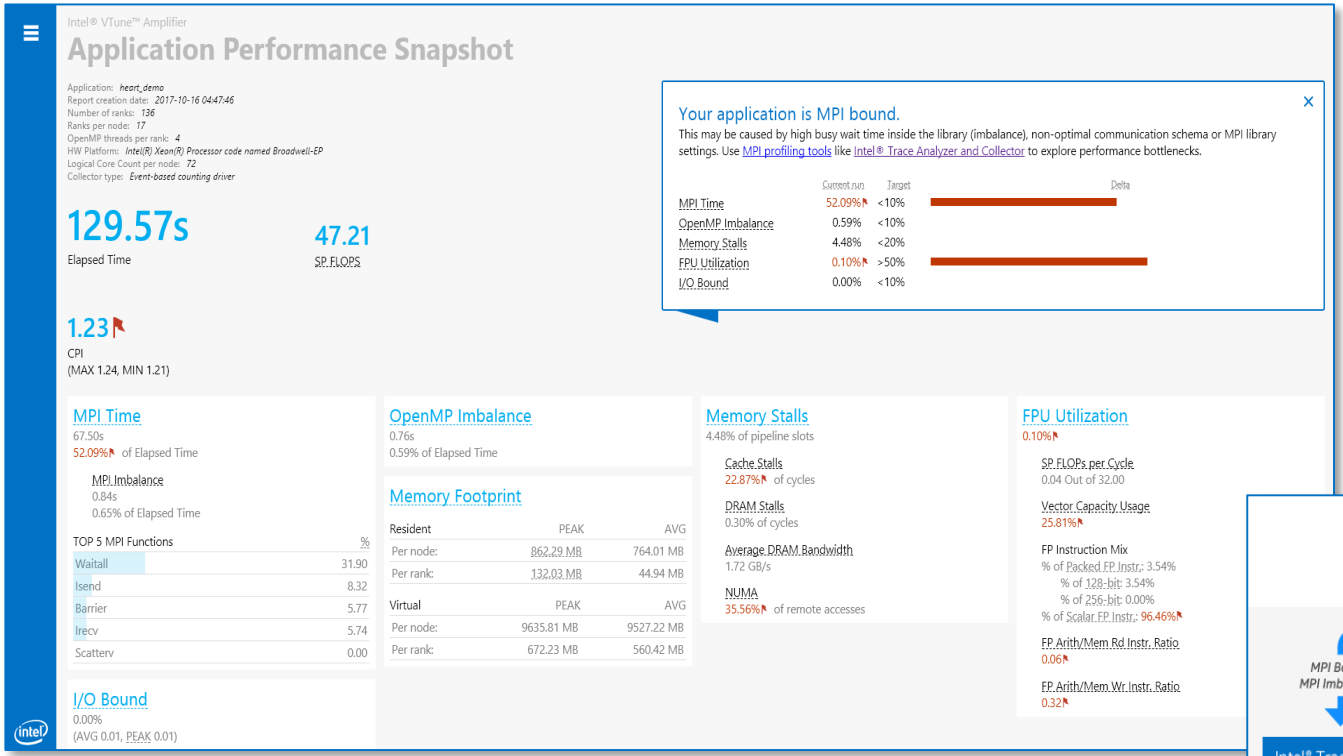
  Tip: use aps “-start-paused” option allows to start application without profiling and skip initialization phase

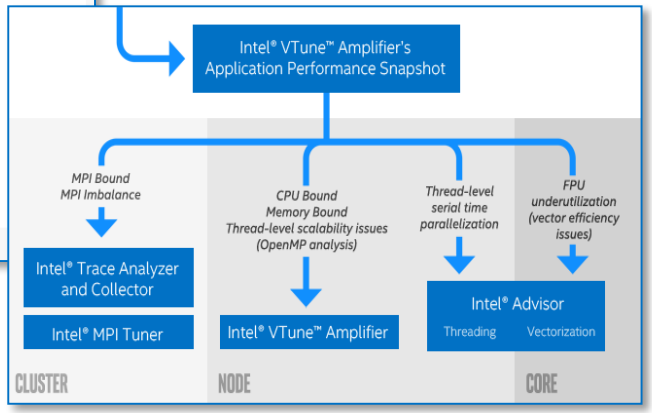
SUMMARY

Intel® VTune™ Amplifier's Application Performance Snapshot is:

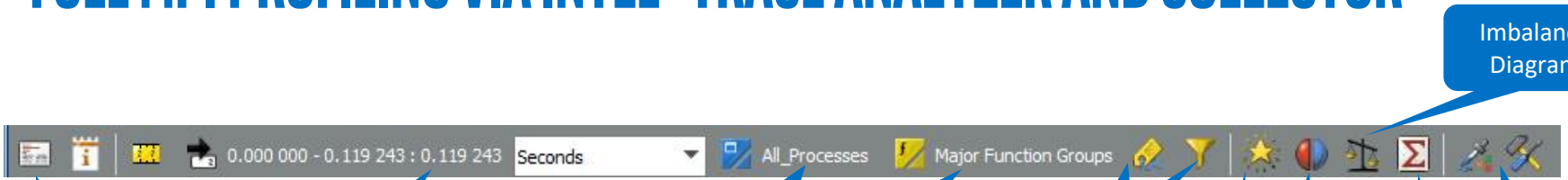
- Your entry point for HPC application performance analysis
- A part of Parallel Studio XE or easy to install and free standalone package
- Simple and well-structured command line and HTML reports
- Clear next steps for tuning with connection to detailed performance tools
- Tool-of-choice of MPI efficiency analysis at scale

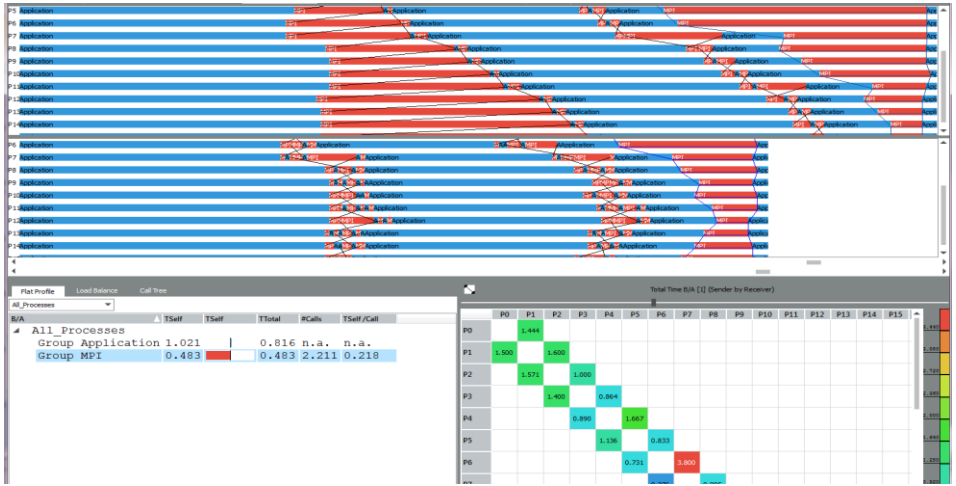
BACK-UP





FULL MPI PROFILING VIA INTEL® TRACE ANALYZER AND COLLECTOR





- Compare 2 communication profiles – focus on bottlenecks
- Shows how MPI processes interact

IMPROVE PERFORMANCE OF MPI + OPENMP APPLICATIONS

ENHANCED MPI METRICS FOR HPC APPLICATION ANALYSIS

Threading: CPU Utilization

- Serial vs. Parallel time
- Top OpenMP regions by potential gain
- Tip: Use hotspot OpenMP region analysis for more detail

Memory Access Efficiency

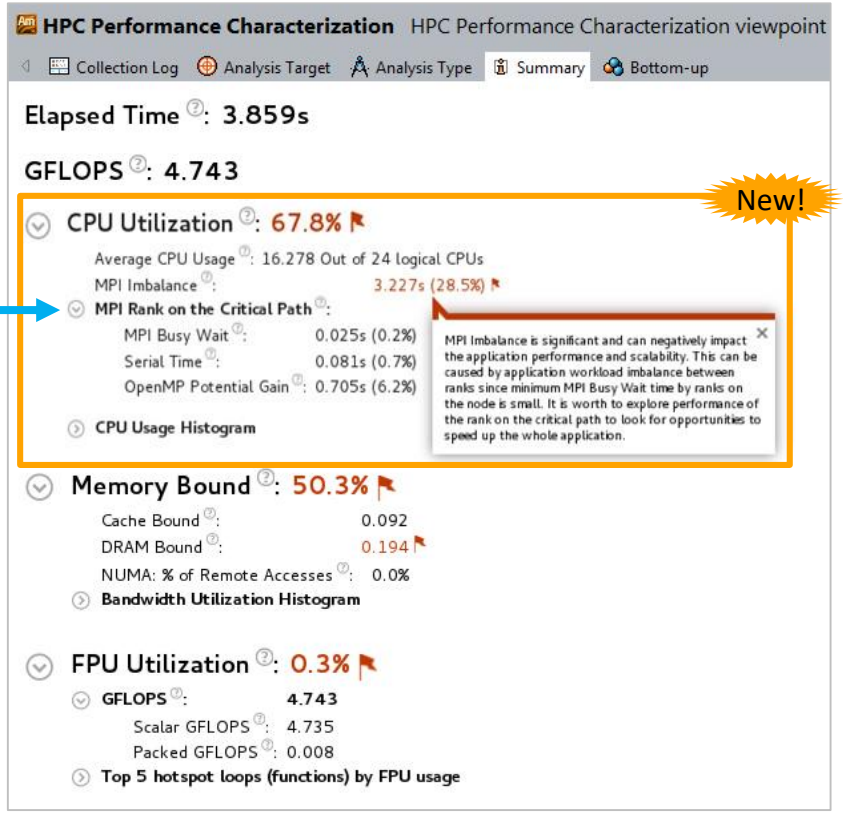
- Stalls by memory hierarchy
- Bandwidth utilization
- Tip: Use Memory Access analysis

Vectorization: FPU Utilization

- FLOPS † estimates from sampling
- Tip: Use Intel Advisor for precise metrics and vectorization optimization

MPI Imbalance Metric

- Metric for performance of rank on critical path
- Computational bottlenecks and outlier rank behavior now available in VTune Amplifier
- For communication pattern problems between ranks use Intel® Trace Analyzer and Collector (ITAC)



† For 3rd, 5th, 6th Generation Intel® Core™ processors and second generation Intel® Xeon Phi™ processor code named Knights Landing.

New!

Find Effective Optimization Strategies

Intel Advisor: Cache-aware roofline analysis

Roofs Show Platform Limits

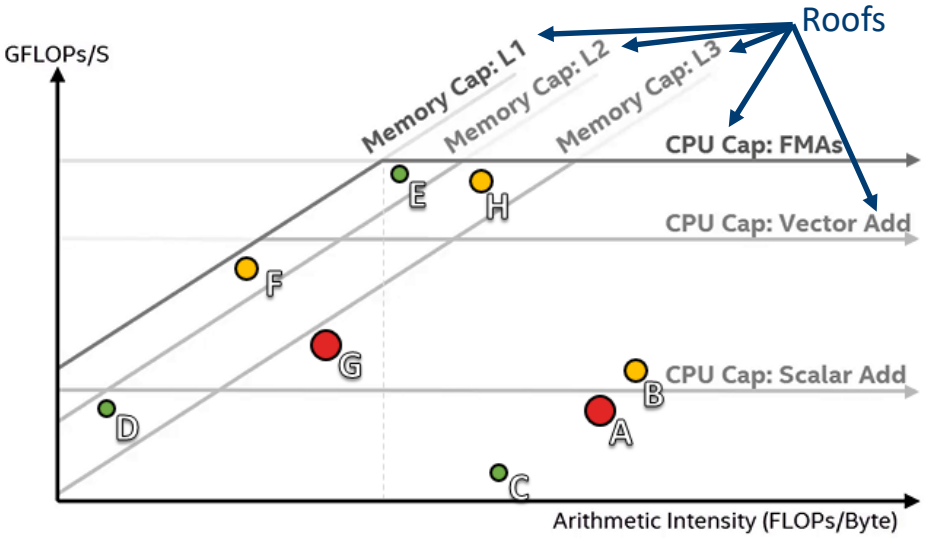
- Memory, cache & compute limits

Dots Are Loops

- Bigger, red dots take more time so optimization has a bigger impact
- Dots farther from a roof have more room for improvement

Higher Dot = Higher GFLOPs/sec

- Optimization moves dots up
- Algorithmic changes move dots horizontally



Which loops should we optimize?

- A and G are the best candidates
- B has room to improve, but will have less impact
- E, C, D, and H are poor candidates

Roofline tutorial video

Legal Disclaimer & Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Optimization Notice

Intel’s compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804


intel
Software